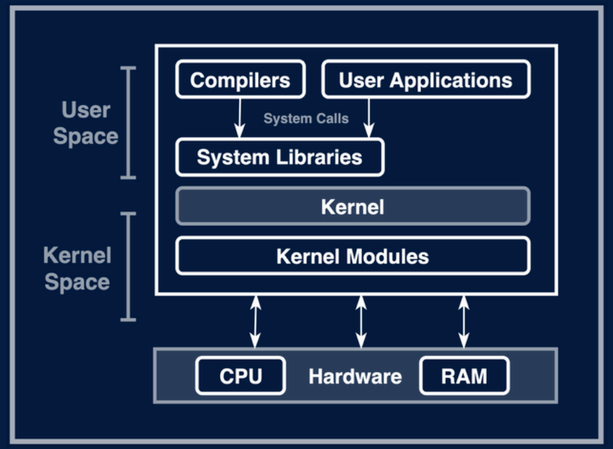
Kernel and User Modes



A computer operates in two modes:

1. User Mode – Limited access to system resources. (user applications and non-critical processes run)
2. Kernel Mode – Full access to system resources. (the core of the operating system (kernel) runs)

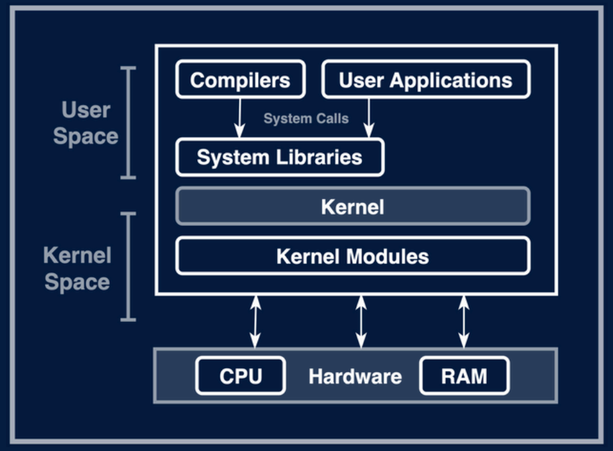
Lets understand from an example : Think of a restaurant:

- **User Mode:** Customers (apps) can only place orders (requests) but cannot enter the kitchen. In the same way apps can only place requests
- **Kernel Mode:** Chefs (OS) have full access to ingredients (hardware) and can prepare food (handle system operations). In the same way OS has full access to hardware and can handle system operations.

A customer cannot cook their own meal (direct hardware access); they must request the chef to do it (system call). In the same way user doesn't have direct hardware access so they request the OS via system call.

Kernel and User Modes

Mode bit : The Mode Bit is a special flag (binary bit) in the CPU that indicates whether the system is running in User Mode (1) or Kernel Mode (0). It plays a crucial role in ensuring security and stability by restricting direct access to critical system resources.



Current mode : Kernel-0, User-1

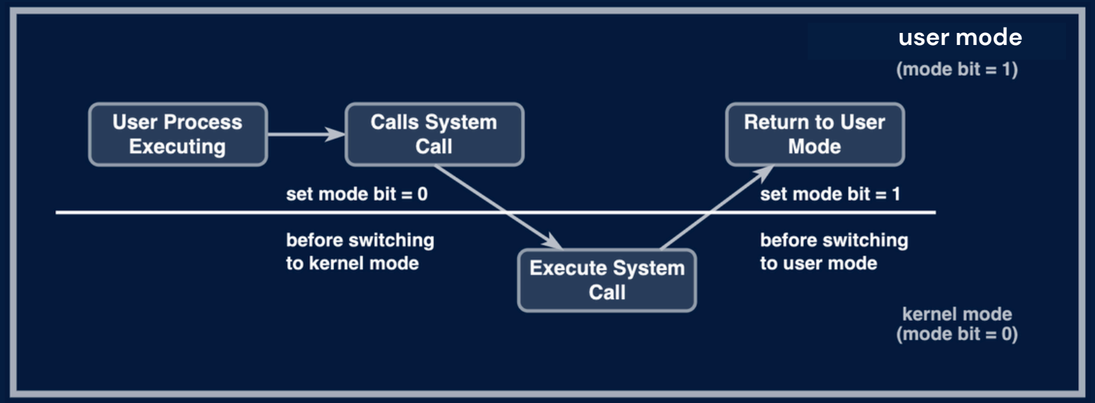
For ex : You open a file in Notepad.

- Notepad (running in User Mode) requests the OS to open a file.
- The CPU sets the Mode Bit to 0 (Kernel Mode) to allow the OS to access the disk.
- The OS reads the file and hands control back to Notepad.
- The CPU sets the Mode Bit back to 1 (User Mode), ensuring restricted access.

Kernel and User Modes

- **Kernel Mode** is the mode in which the core of the operating system (kernel) runs. It has unrestricted access to all system resources (hardware, memory, I/O devices, etc.). The OS kernel executes all privileged operations (e.g., memory management, process scheduling, hardware access).

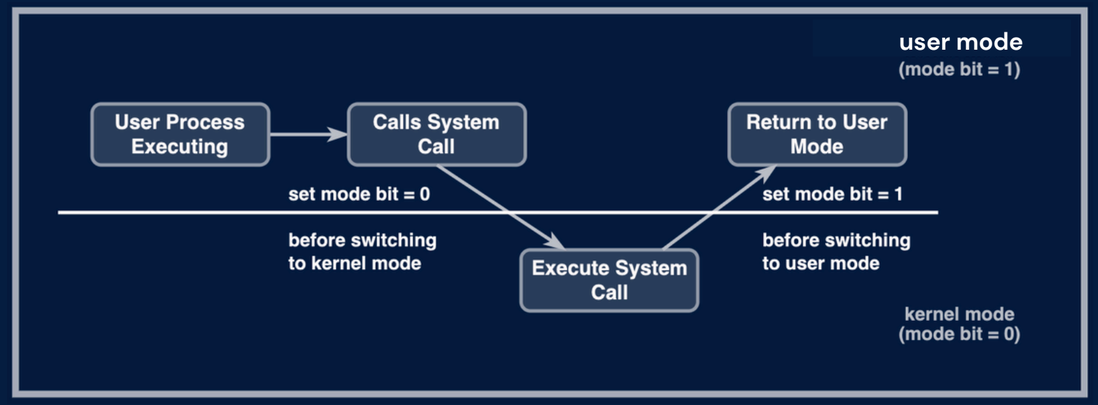
- Kernel Mode is highly privileged because it can access and modify system-critical resources. Malfunctioning code in kernel mode can lead to system crashes or security vulnerabilities. Therefore, only trusted OS components (like system services) run in kernel mode.



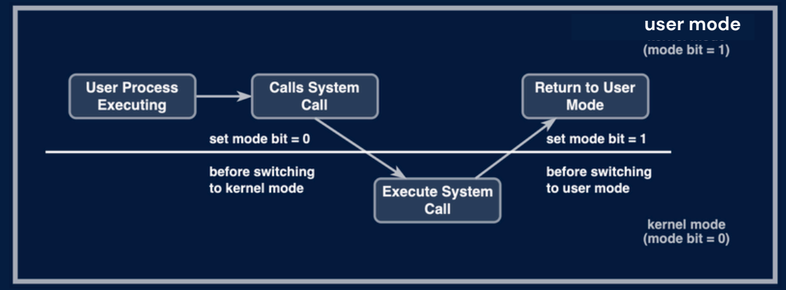
Kernel and User Modes

- **User Mode** is the mode in which user applications and non-critical processes run. The OS ensures that user applications have restricted access to system resources and cannot directly interact with the hardware.

- User Mode acts as a protective barrier, ensuring that user applications cannot directly harm the system or gain unauthorized access to critical resources. Even if a user application is compromised, it cannot take full control of the system.



Kernel and User Modes



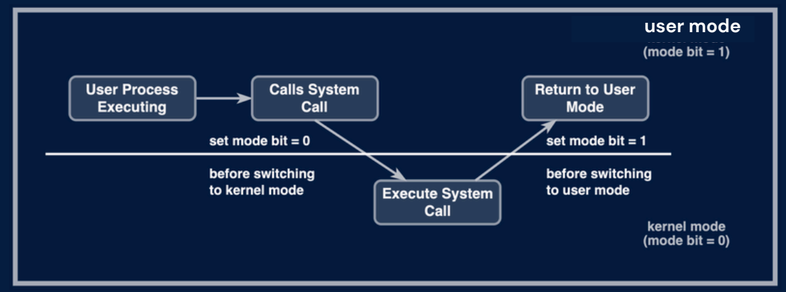
The transition between these two modes is called a **context switch** and happens when:

- **System Calls**: When a user program needs to access hardware or perform a privileged operation, it makes a system call to request this from the kernel. The OS switches from user mode to kernel mode to handle the request.

- **Interrupts**: Hardware or software interrupts trigger the kernel to execute an interrupt service routine in kernel mode. After handling the interrupt, control returns to user mode.

- **Exceptions**: If a user program encounters a critical error (like dividing by zero), an exception may cause a transition to kernel mode for error handling.

Kernel and User Modes



Imagine you want to print something on the screen using a program.

Step-1 : User Mode (Your Program Starts Execution)

- The printf() function runs in User Mode.
- It cannot directly access the screen (hardware).

Step-2 : System Call (Request to Kernel Mode)

- printf() internally calls the write() system call.
- This triggers a trap/interrupt, which tells the CPU to switch to Kernel Mode.

Step-3 : Kernel Mode (OS Handles the Request)

- The OS kernel processes the write() system call.
- It interacts with the display driver to print text on the screen.

Step-4 : Switching Back to User Mode

- The kernel completes the task and returns control to the program.
- The CPU switches back to User Mode, and the program continues execution.

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Kernel and User Modes

In **Kernel Mode**, the OS has complete control over the system and can perform high-level operations, while **User Mode** limits access to system resources, providing a secure environment for running applications. This separation ensures that the system remains stable and secure while allowing applications to operate effectively.